Routing Information Protocol aka (let ‘er) RIP

IP Routing

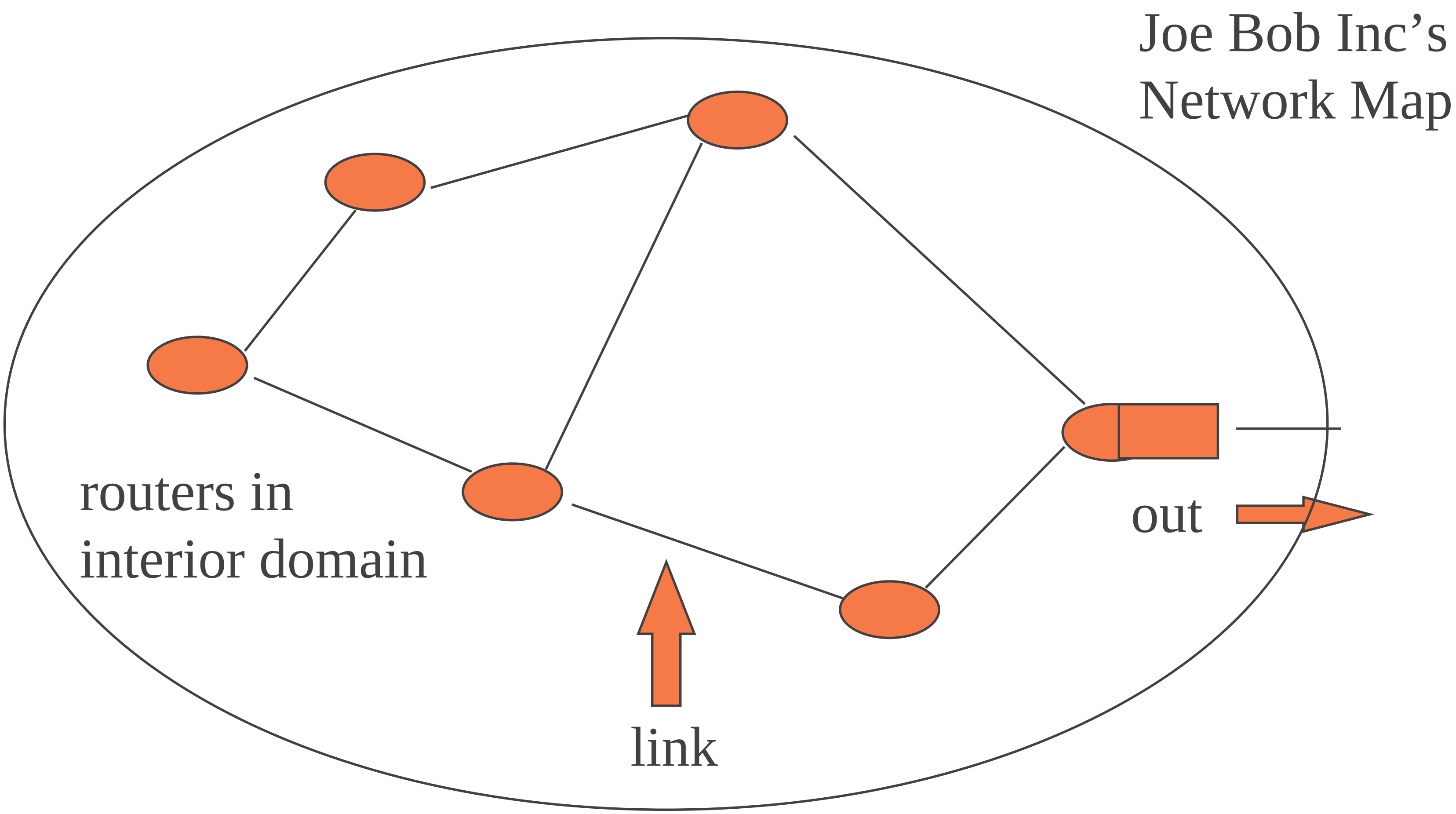
outline

- intro
- theory including convergence and bugs
- rip v1 protocol
- rip v2 protocol
- Cisco config example with default route redistribution
- conclusions

protocols acc. to topology

topology	IETF	ISO/OSI
intra-link	ARP	ES-IS
intra-domain	RIP, RIP(2), OSPF	IS-IS
inter-domain	EGP, BGP(4)	IDRP, IDPR

the Interior - RIP or OSPF



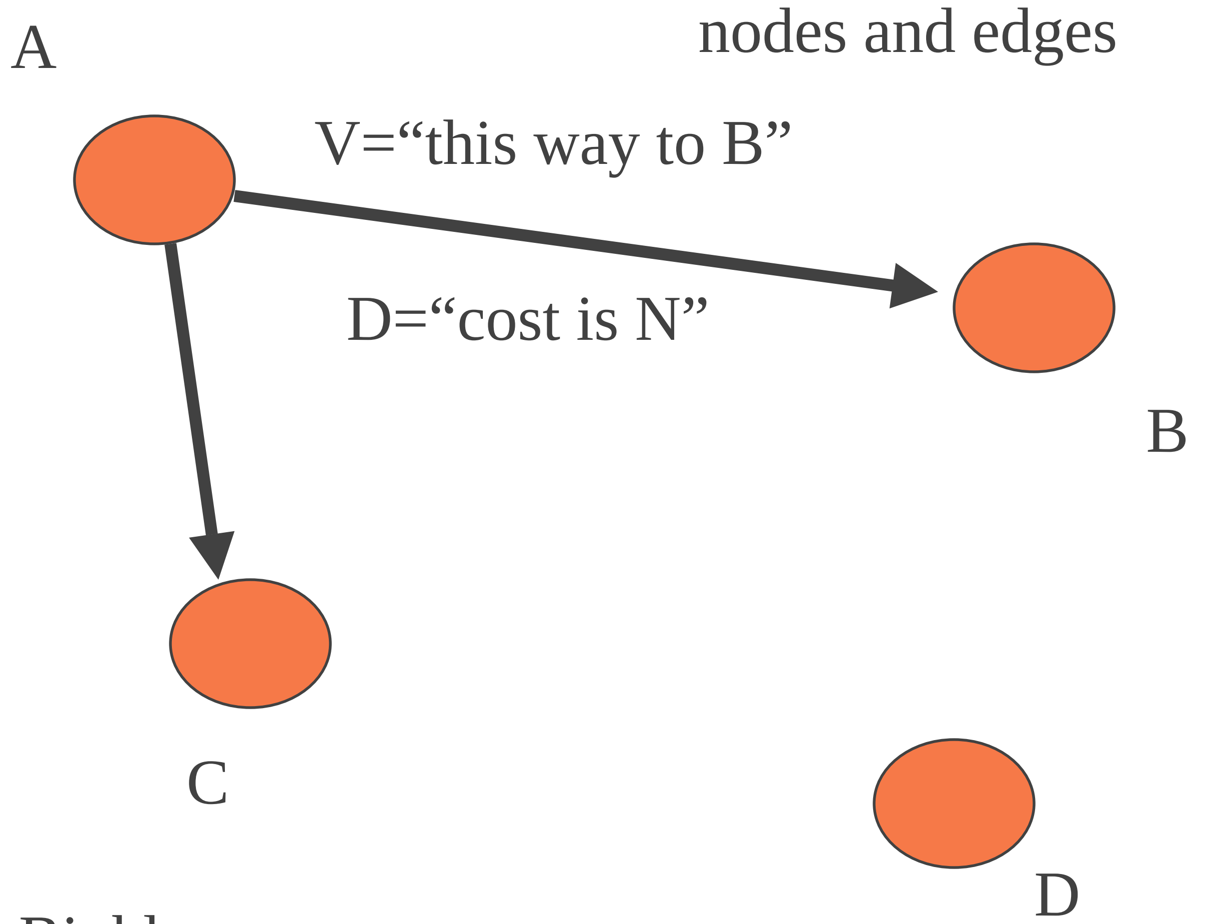
bibliography

- RIPv1, RFC 1058, Charles Hedrick, 1988.
  - documented existing practice
- RIPv2, Gary Malkin, RFC 1388
  - RIPv2, RIP speaks CIDR (netmasks included with destination)
  - RFC 2453 is update, 1389 MIB, 1721-1724
  - MD5 authentication, 2082
- Huitema, Routing in the Internet, 2nd Edition, 1999

history

- Bellman/Ford/Fulkerson and Distance/Vector idea, late 50's, early 60's
- Bellman, "Dynamic Programming", Princeton University Press, 1957
- Vector-Distance can mean IP Destination/Hop-Count (as with RIP)
- Distance in other protocols might mean something else
  - hello, TIME; BGP, A.S. path to destination

Vector-Distance



cont.

- BSD app based on XNS (Xerox) version, Netware RIP is similar too (surprise)
  - BSD 4.2 on VAX (1982 or so)
- done first and RFC 1058 (1988) later created
- in widespread use for at least two reasons
  - widely available, came with that there Sun WS
  - # routed & is (mostly) all you need to do
- BSD routed and Cornell gated support it (free)
- Cisco evolved into IGRP, and later EIGRP
- Appletalk - Routing Table Maint. Protocol (RTMP)

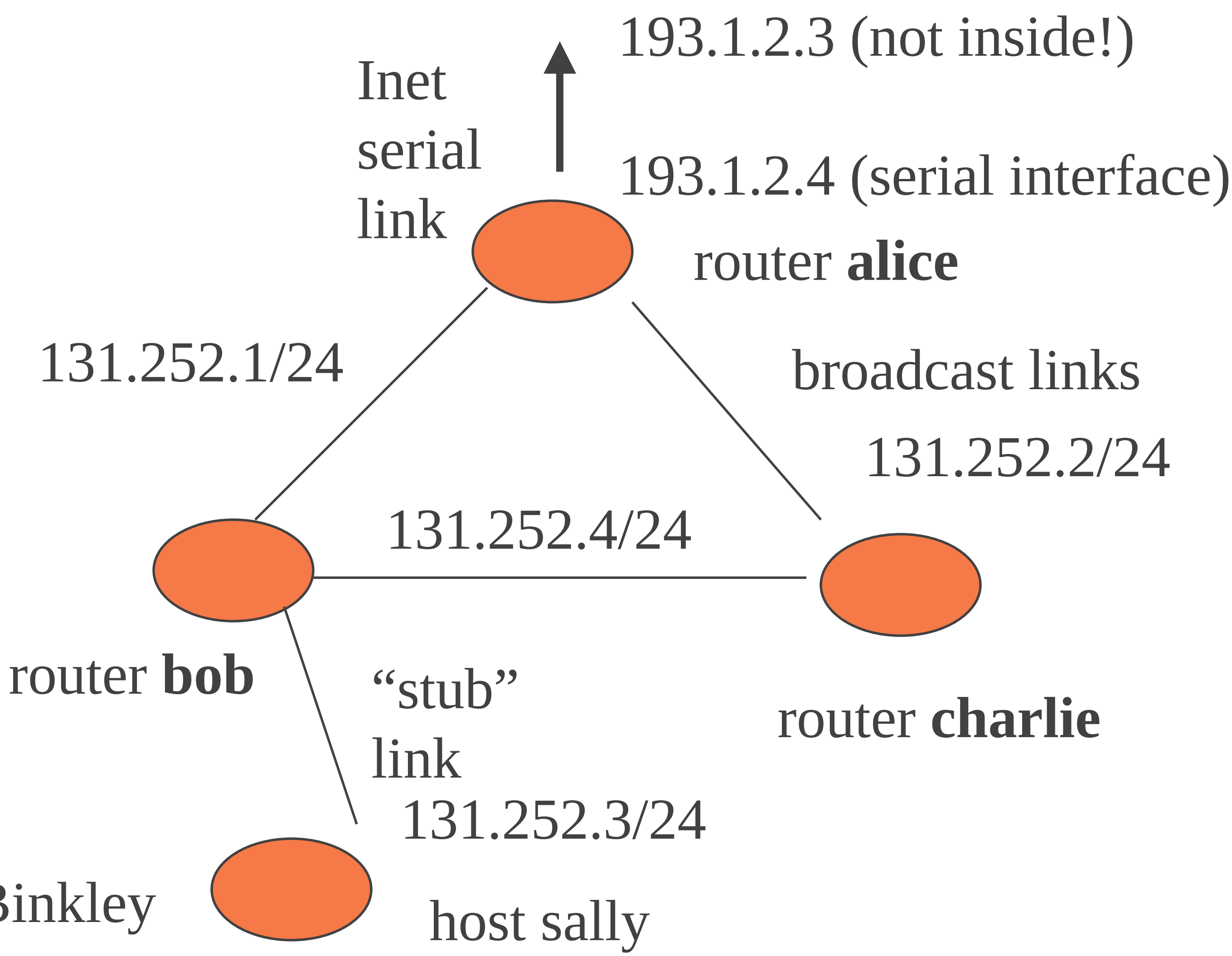
RIP details

- messages carried in UDP datagrams, send/recv on port 520
- broadcast every 30 seconds, routing table as pairs of (to net, hop count) e.g., v1 ip dst = 255.255.255.255
- **hop count, direct connect == 1, network one router away is 2 hops away**
- new route with shorter hop count replaces older route
- on init, router requests route table from neighbors
- therefore two fundamental message types
  - request (done at boot. give me your routing table)
  - **response** (almost all messages are response)

more RIP details

- when routing response received, routing table is updated (metrics aren’t typically displayed in netstat -rn unfortunately)
- route has timeout. 3 minutes, no new info, then mark with metric=16, one minute later delete (**holddown** so the fact that route is gone is propagated)
- infinity == 16, RIP can suffer count to infinity
- default route is route to 0.0.0.0
- routers are “active”, hosts are “passive”, determined by whether or not system > 1 i/f (can set by hand)

consider simple Interior domain



traditional UNIX workstation as router - configuration

- overly simplified ...
- router alice (border router)
  - # routed -g + static route to outside Inet
- router/s bob and charlie
  - #routed
- random workstation (not router):
  - #routed -p (passive mode, won't send)

points to ponder

- border router MIGHT have static route on serial link
  - ideally might NOT want to RIP out that i/f and waste bandwidth, annoy ISP/router neighbor
- border router sends (0.0.0.0,1) default to neighbors who can propagate to hosts
- misbehaving host might fireup
  - #routed -g
  - bring down part/all of net
  - routers need to ignore hosts in terms of **routing filters**

study questions

- with UNIX rip, how can a border router NOT send rip update out serial i/f ?
- with Cisco rip, how can a border router NOT send rip update out serial i/f ?
- with a sniffer (say tcpdump) how can we watch rip updates only ?
- what value is there to a host if it runs RIP, but only has one link router ?
  - what if the host has two interfaces?
- at sally, what is the value of the default metric?

theory

- neighbors in mesh **“tell the neighbors about the world”**
- i.e., they periodically broadcast their “routing table”
- v1 routing table actually pairs of **(ip dest, hop count)**
- directly connected network has hop count of 1
- infinity (unreachable) is 16, 15 maximum hop count
- hosts may listen but don’t broadcast
  - can learn default route dynamically
  - can learn paths to other networks if redundant routers

simple theory

- writer:
  - every 30 seconds send out
    (131.252.1.0, 1)
    (131.252.2.0, 2)
    (0.0.0.0, 3)
- reader:
  - read broadcast and merge with routing table
  - add new tuples, or modify hop-count for existing tuples possibly including next-hop

timers

- **deletion**: for each new tuple, start timer, toss if no refresh in 180 seconds
  - note: N times broadcast (6 * 30)
  - if we don't hear from you (which can happen due to collisions, noise, etc.) we forget about you
- **write timer**: resend every 30 seconds
- **garbage timer**: (Cisco), advertise with unreachable (16) for 60 seconds before deletion
- **holddown**: if update has higher hop count, don't forward for 180 seconds (delay bad news)

RIP and control theory

- remember chicken and egg problem of routing; i.e.,
  - in absence of routing, how does routing itself work?
- rip v1 relies on UDP/IP broadcast 255.255.255.255 (v2 uses multicast)
  - application-layer flooding, no ACKS
  - in one interface and out the others (er, all , actually)

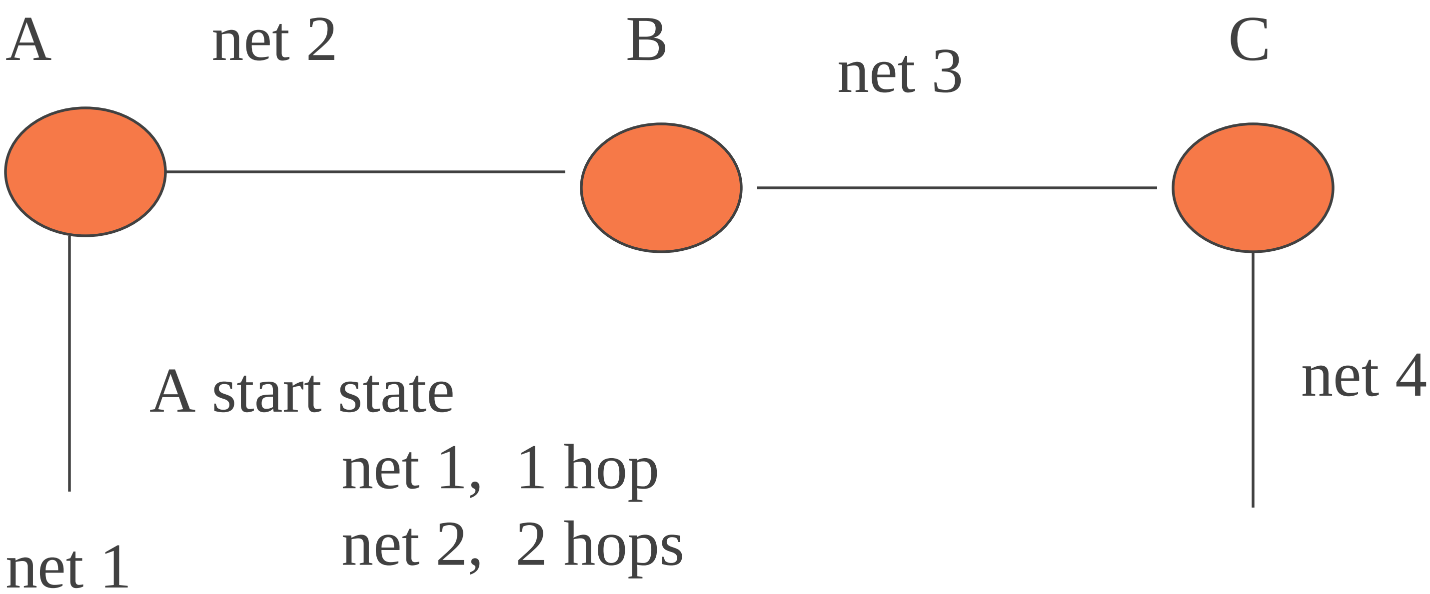
control theory, cont.

- bottom line: RIP relies on neighbors being directly connected
- takes advantage of broadcast on media like ethernet
- broadcasts are resent at a rate greater than deletion timer (N broadcasts before tuple is deleted in routing table)

network states

- **start** (router or entire network) - initial routing table (direct connects)
- **linkdown** or linkup
  - loss of router is extreme case of this
- **convergence** (steady state)
  - start or link change must lead here
- ***convergence means routers have same destinations, possibly different metrics***

define convergence

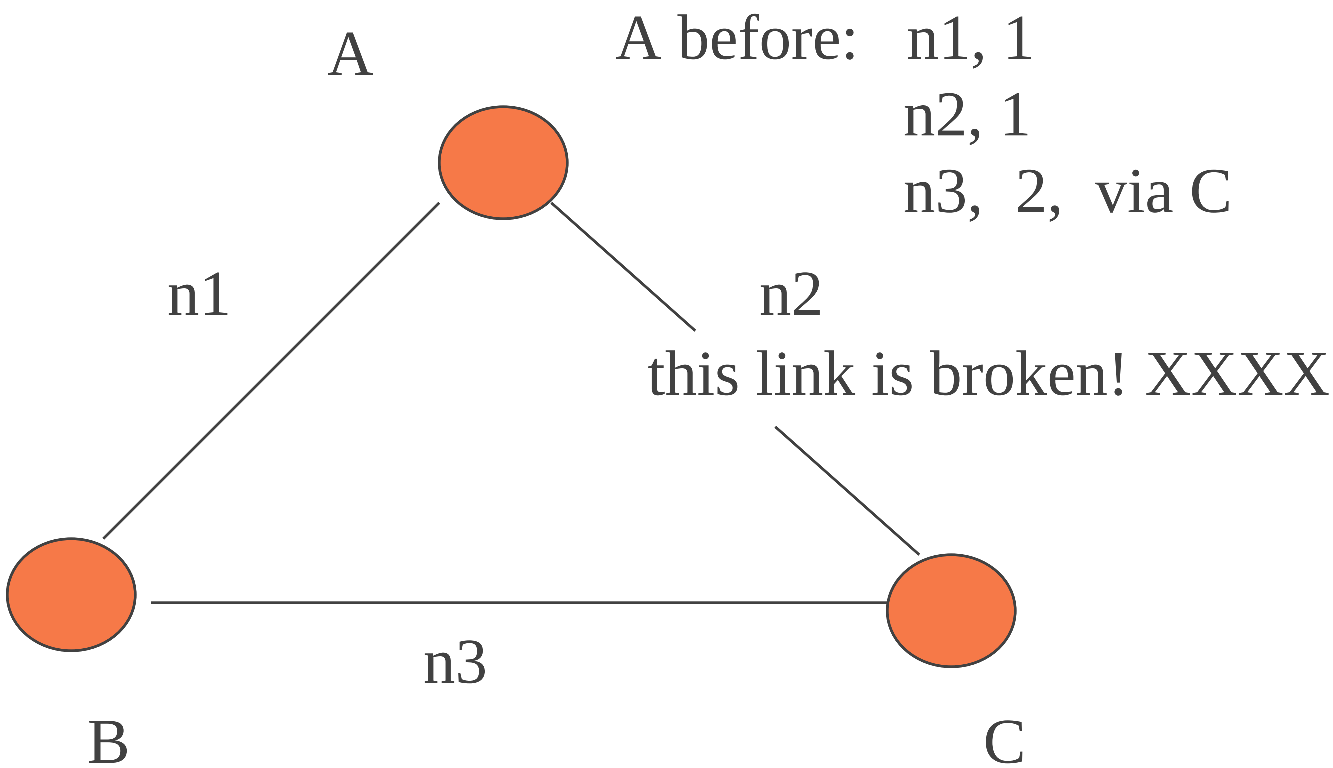


A start state

net 1, 1 hop
net 2, 2 hops

how many broadcasts before convergence
what do routing tables, A, B, C look like?

link down



A/C just crashed. What has to happen for convergence?

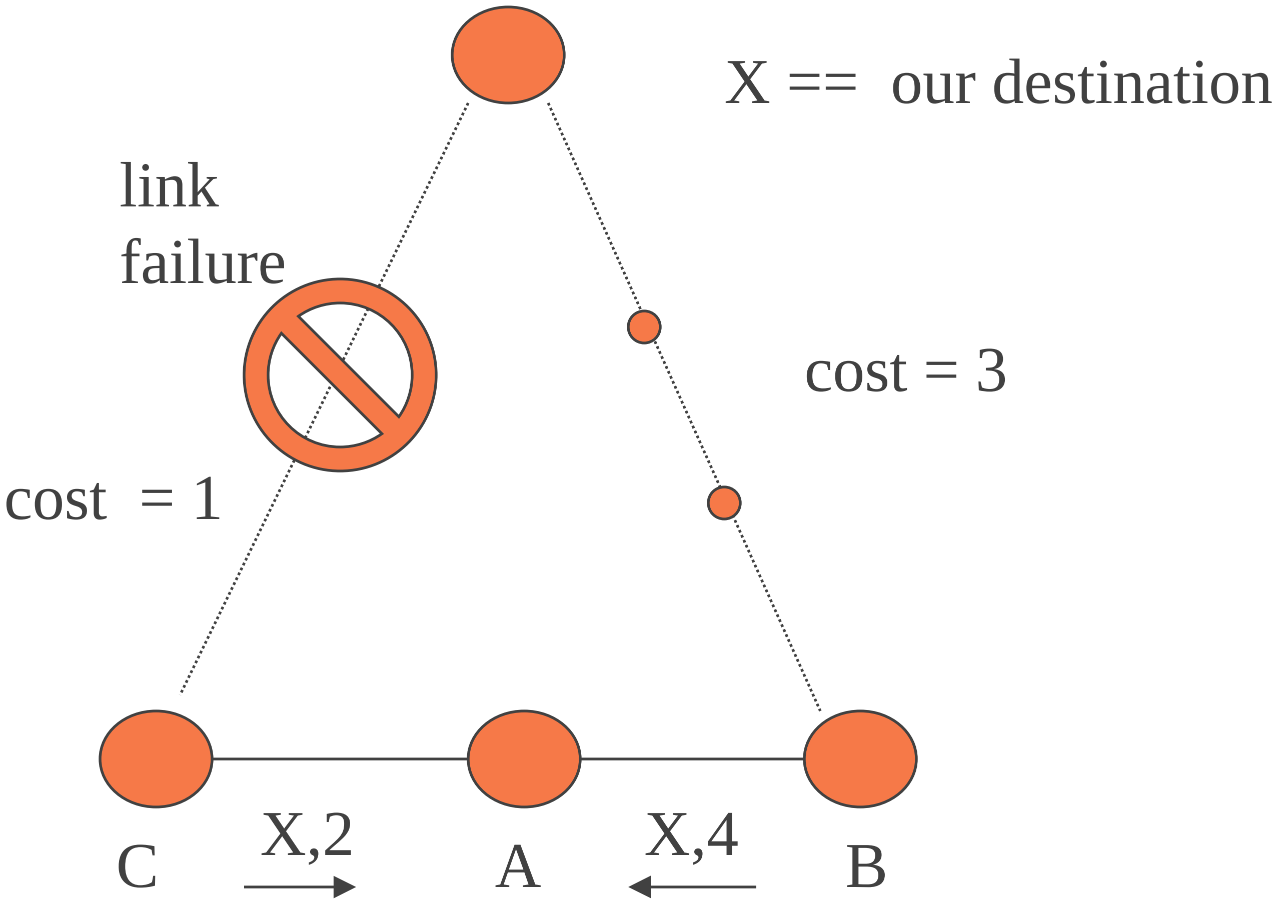
how can A learn the link to C failed ?

- 1. ideally, because A has a link-layer sub-protocol that will tell it the A/C link failed
  - no such beast with ethernet - possible if A's i/f fails
- 2. worst-case, A's tuple for C times out
- when A has learned that C does not exist, it will then believe B has a path
  - to C, via B, 2 hops
- note the benefits of the previous redundant mesh (compared to previous non-redundant example)

how could a link go down?

- Backhoe (link wire cut)
- interface card blows up
- router blows up
- variations on “backhoe”
  - chair runs over ethernet cable on floor 1 too many times ...
  - sys admin kicks AUI ethernet cable out of workstation and doesn’t notice
  - you didn’t purchase the UPS after all?

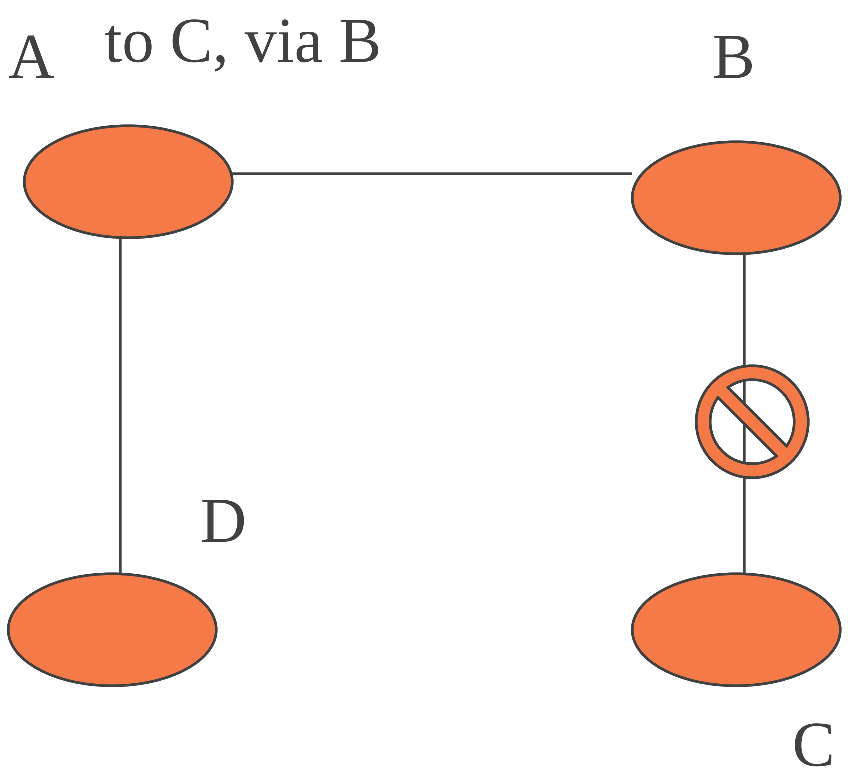
bouncing effect



bouncing leads to data pkt loops

- X to C link fails
- unfortunately A tells C that it has tuple (to X, cost=3) before C can tell A that link is down
- A will tell C cost to X is (X,4)
- point is that it MAY take awhile for A to discover that path thru B is "better" (and real!)
- **note that data packets thru A for dst=X will be caught in a loop (IP ttl is a good idea ...)**
- **2 router black hole**

count to infinity



assume B/C link crashes

16 is a very small infinity

- A knows to C, 2 hops, via B
- B has direct connection to C, knows C is down
- before B can tell A, A tells B
  - to C, 2 hops!
  - B believes A. Joy!. A knows how to get to C!
- B tells A, to C, 3 hops
- A believes B (after all B is the way to C)
- count up to 16 before giving up
- is this likely? (murphy's law, bad news is faster)

the count of Monte RIP OFF (pun)

- infinity must be small
- limits the routing diameter, therefore scalability limitation
  - not important one though
- note two cases:
  - C is temporarily cutoff due to bouncing effect, but redundant path exists
  - no redundant path, packets will loop until infinity, at which point routers can return ICMP destination unreachable

important basic idea:

- routers can do 1 of three things:
  - 1. actually correctly route packet
  - 2. get packet, not have route, and send ICMP destination unreachable
    - » imperfect, but much better than
  - 3. get packet, and “lose it”; e.g., packet stuck in routing loop until TTL timeout
    - » no ICMP unreachable
    - » sometimes routers need to “sink” packets

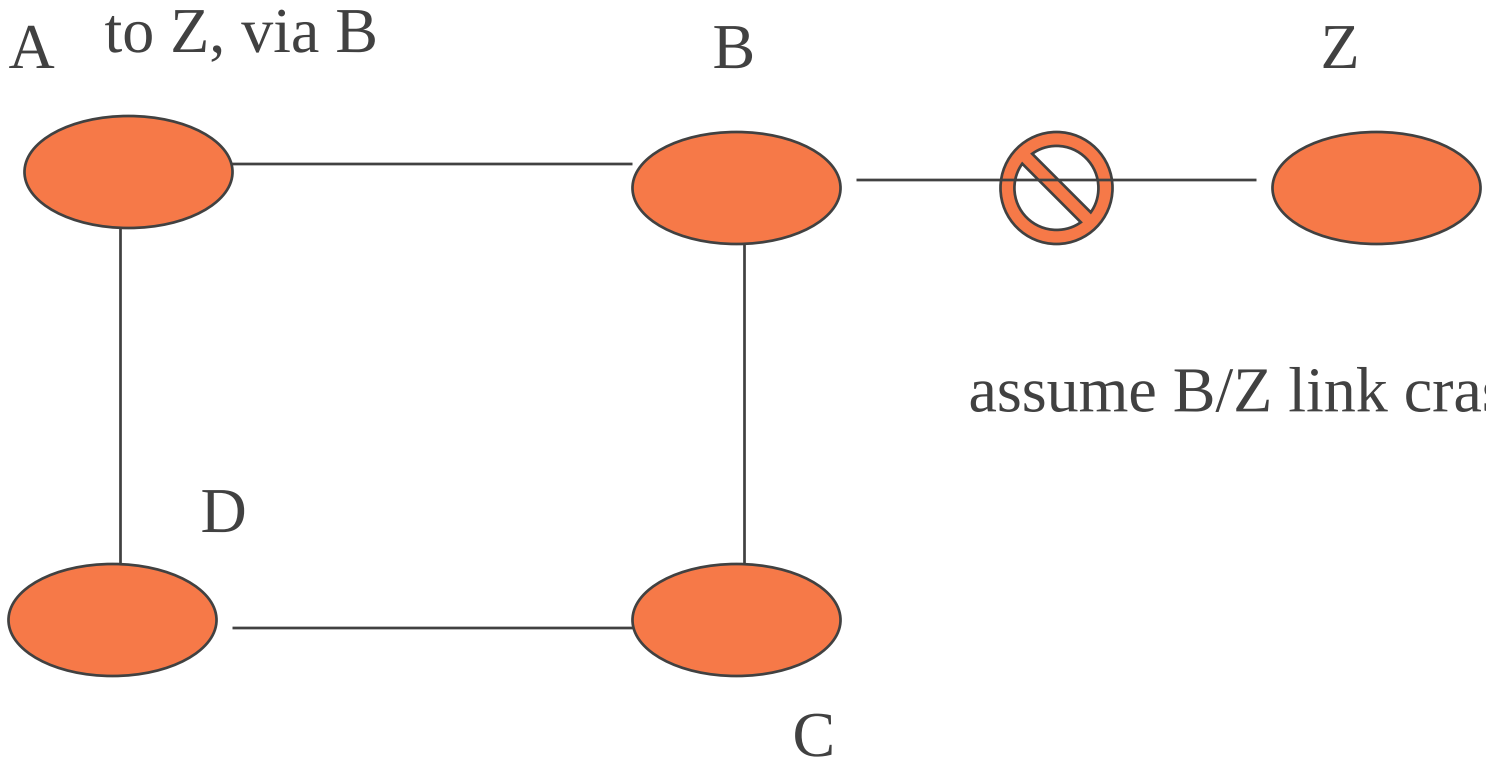
count to infinity/bouncing effect

- summary: count to infinity can cause or exacerbate convergence time
  - slow convergence is possible result
- various imperfect fixups exist
  - split horizon
  - triggered updates
  - holddown
- of course, complete routing map can cure this problem (EIGRP or OSPF)

split horizon

- **split-horizon**: keep track of interface thru which update came
- two ways to do this: (A to B to C)
  - 1. A to B, does not include C
  - 2. A to B, includes C with metric set to 16, this is "**poison reverse**" explicit negative update
- poison reverse basically: "whatever you may think, I am not the path ..."

split horizon bug



assume B/Z link crashes

D can still tell C it knows the path to Z (thru A)

triggered update

- remember we have at least a deletion timer (180 seconds + possible holddown time)
  - and a write timer (30 seconds)
- if we discover failure (our own link failed or we have another clue) (or any change)
  - immediately send new information. MAY send only that information (changed tuple/s)
  - not wait for write timer or deletion timer
  - call this **triggered update**

pros/cons

- pros: may hopefully speed up convergence
  - or speed up count to infinity...
- cons:
  - 1. we could spend all of our time processing triggered updates
  - 2. might trigger broadcast storm
- may hold down frequency of triggered updates; i.e., 1..5 seconds per update

holddown

- since it is likely that bad news may travel faster (we need a name for the opposite of Murphy; i.e., good luck)
- Cisco routers use **holddown** mechanism
- in this case, this means if recv. metric > current metric, may wait a bit
- if we are lucky, we might obviate count to infinity (or make slow convergence worse?)

point/s to ponder

- given fixups for count to infinity/slow convergence problems
- is RIP still so simple?
- RIP is truly: **routing by rumor**
- **pssst... I know the way to Z**
  - well, actually Bob told me the way to Z
- the protocol has some protection against routing loops -- but not much

RIP v1 encapsulation

ip src = X ip dst = 255.255. 255.255	UDP src/dst =520	RIP header + tuples

RIP(1) header

command	version(1)	must be zero
family(2)		must be zero
ip address		
must be zero		
must be zero		
metric: (1-16)		

one route entry (rows from family(2) to metric)

up to 24 more routes, 25 routes max (< 512)

note: command: 1, request; 2, response

RIPv1 details

- UDP packets limited to 25 routing table entries, 512 bytes
- if more entries, send more broadcast packets
- consider 131.252.222.16/28 - you can’t tell if this is network (subnet) or host
- you only have subnet masks bound to local interfaces
- 0.0.0.0 means default route, 16 means NO!

RIP header

- command = 1: request, 2: reply
  - typical write/update is reply (even if no request)
- version: 1 of course
- address family + 4 bytes of zero + IP addr + 8 bytes of zero + metric == 1 tuple
- 20 bytes per tuple
- hope was other protocols might use but didn’t happen

rip request

- may be sent at router boot or link boot to request routing table from neighbor
- actually two forms
  - 1. request full listing
  - 2. request specific route (debug software)
- full listing format: address family == 0, address is 0.0.0.0, metric=16
- command = 1 (of course)
- reply is unicast to request (think of BSD recvfrom(2) for how to get IP addr of peer)

message processing algorithm

- read message
- do sanity checks
  - make sure IP address not loopback/broadcast
  - make sure metric in bounds
- increase metric by 1
- search routing table by destination
- if entry not found and metric not infinite
  - set ip dst, next hop ip, interface, metric
  - start 180 second delete timer
  - store new route in ip-layer routing table

algorithm, cont

- if recv. metric better than current metric
  - delete old entry
  - store new entry and restart timer
- if we find entry and sender is current next hop and sender's metric changed
  - change our metric, restart timer
- this is not complex enough -- e.g., have to consider triggered updates

implementation note

- e.g., on UNIX with routed
  - routed contains an application-level routing table
  - this is NOT the kernel routing table
- not unusual for their to be an update-oriented table (a routing update database)
- which may contain redundant information not stored in kernel ip routing table

e.g., consider

- we are A and we have two paths to C that have equal weights
  - routing database therefore:
  - to C, via B, 2 hops
  - to C, via D, 3 hops
- we store the route via B in our routing table and use that
- a smarter implementation may be able to use the redundant information

RIP(2) header

command	version(2)	routing domain
family(2)		route tag
ip address		
net mask		
next hop IP address		
metric: (1-16)		

one route entry (family(2) through metric)

up to 24 more routes, 25 routes max (< 512)

RIP-2

- RFC 1388 (1993)
- zero fields cleverly used, should interoperate if RIP(1) ignores fields
- version is 2
- routing domain can be used to allow more than one RIP domain on a campus; more than one routed on a system
- route tag - AS number, communicate boundary info (not used by RIP)
- subnet mask - for CIDR, route == (ip, net mask)
- next hop, ip address for VIA part of route (as opposed to getting it from IP src)

RIP-2

- clear-text password
  - better authentication exists
- can use multicasting as opposed to broadcast, thus hosts that
  - “don’t give a RIP(2)” can ignore it
- send to 224.0.0.9 (all-ripv2-routers)
- remember multicast range 224.0.0.1 to 224.0.0.255 are not forwardable and for routing only ...

RIPv2 routing protocol security

- possible dangers: man in the middle attack OR denial of server (DOS)
- MITM means somebody reroutes packets to an intermediate host for laundering
  - inject routes into routing table
- DOS - means they just fill you up with junk
  - possible to distract from a real entry attack on some unfortunate victim host

conventional wisdom

- authentication is enough - don’t need to encrypt routing
  - especially within an IGP
- key management means we are likely to store keys in router NVRAM
  - chicken&egg problem for how router gets to access complex backend key database server

v2 authentication

- began with plaintext password
  - of course, spoofable if sniffed
  - possibly useful though to distinguish administrative zones or
  - prevent misconfigured linux host from taking down network
- RFC 2082 - MD5 shared secret authentication

authenticated RIP request format

cmd = 2, vers = 2, etc. — 1 word command

security header:

addrfam=0xffff, type = 3 (MD5)

pkt len, key id, auth data len (16), seq #

— 20 byte security part

authenticated routing table entries

security trailer:

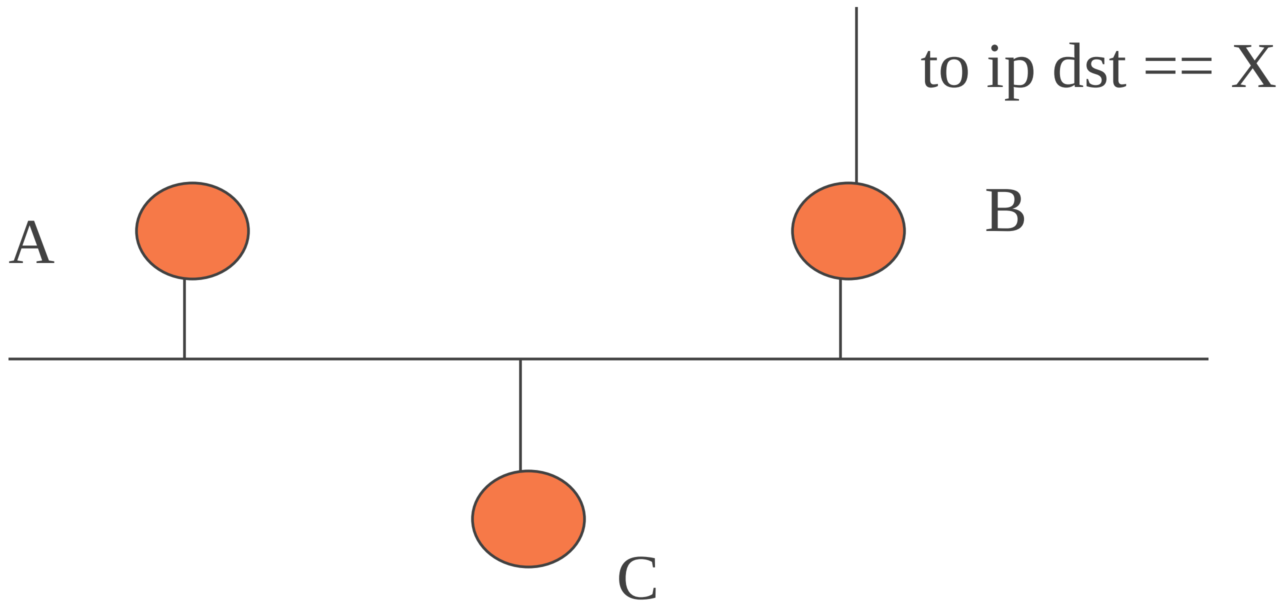
addrfam=0xffff, type = 1

16-byte MD5 “hash” (not the key!)

details

- key ID identifies shared secret key on receiver (MD5 key could be hex 128 bits)
  - e.g., 0xdeadbeefdeadbeefdeadbeefdeadbeef
- sequence number iterated to prevent possibly replay attacks
- authentication mechanism typed as MD5 could be broken, replaced with new stronger version
- key is not sent, merely stored on both sides
- it is more security if per-link, but likely same key per administration zone

next hop is not me (v2 feature)



A can tell C, To X, via B

normally C would infer A as next hop

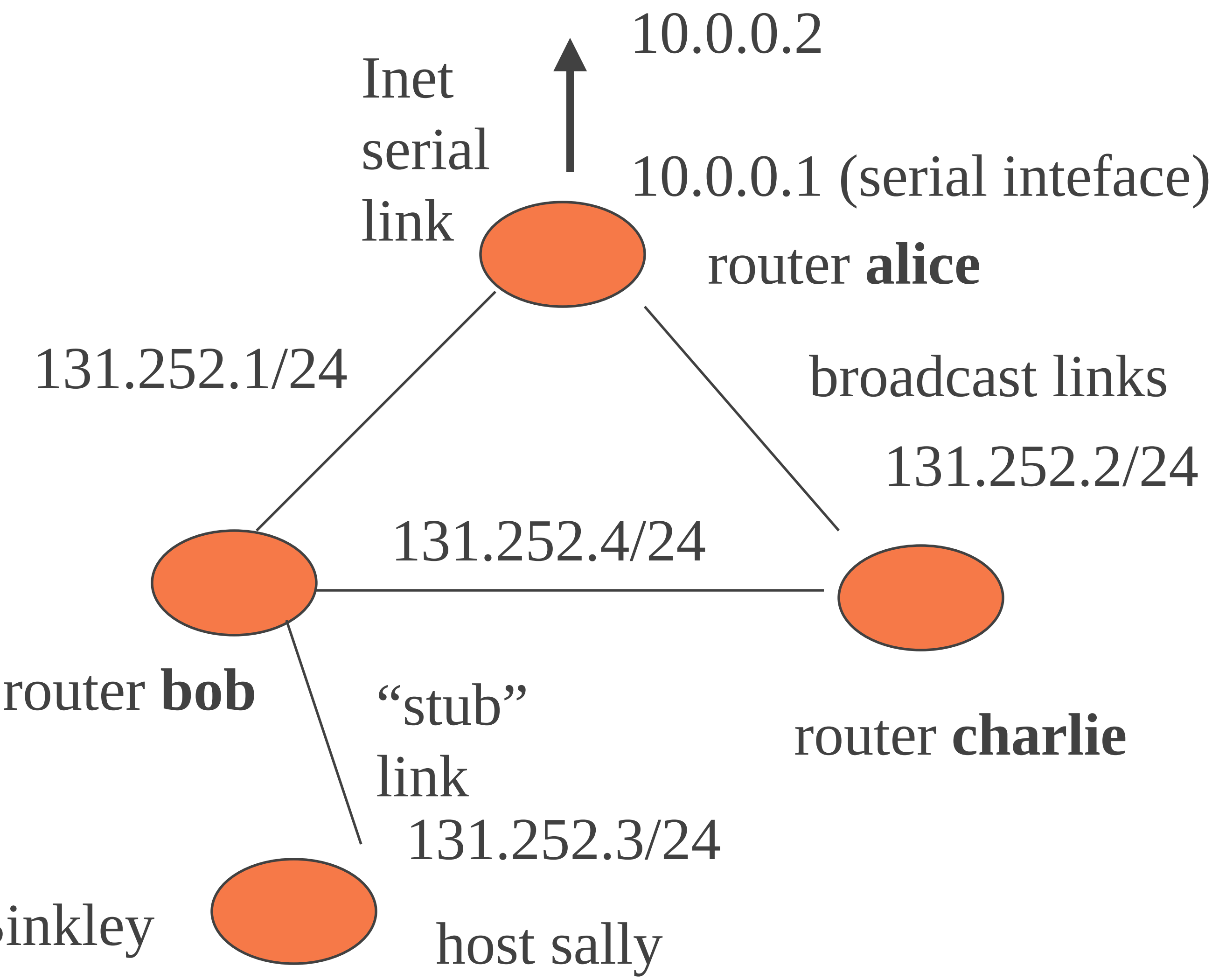
synchronization problem

- Sally Floyd/Van Jacobson 1993/SIGCOMM paper
  - network every 30 seconds was congested
  - RIP routers with no outside timing would self-synchronize and start blasting each others broadcasts and clog up processing broadcasts
  - synchronization caused by implementation choice, result was that broadcast was not random as intended

synchronization problem

- router would fall into chain of
  - 1. receive all router packets now
  - 2. process
  - 3. then send
- over time this caused all routers to fall into same absolute send pattern
- suggestion: randomize update to 15..45 seconds
- this could be generic and widespread problem
- one should engineer-in randomness, not hope for it

consider simple Interior domain



Cisco configuration intro

- router alice
  - router rip

```
  version 1  (or 2)
  network 131.252.1.0
  network 131.252.2.0
  passive-interface serial 0
  redistribute static
  default-information originate
! static route to ISP/router (WAN)
ip route 0.0.0.0 0.0.0.0 10.0.0.2
```

cont.

- bob/charlie simpler
- no static routes
- router rip   (on bob)

  network 131.252.1.0

  network 131.252.4.0

  network 131.252.3.0   (stub network)
- they will pick up and distribute the default route on interior links

possible ways to ignore unwanted updates (assume alice)

- use administrative distance; e.g.,
  router rip...
  distance 255 (this means ignore)
  distance 120 ip-for-charlie
  distance 120 ip-for-bob
- ACL mechanism would work too
  - block RIP on stub interface or subset therein
- or use MD5-based authentication for secure routing protocol updates (best)

conclusions

- "RIP was intended for use in small networks with reasonably uniform technology" - Charles Hedrick
- "DV is routing by rumor"
  - A tells B about C
- RIP not smart by design (UDP analogy)
- OSPF smart by design (TCP analogy)
  - shared idea in OSPF/EIGRP, know topology

to RIP or not to RIP?

- pros
  - simple, stupid... (those are the cons too ...)
  - easy to implement
- cons
  - no understanding of subnetting in v1; e.g.,
    - 121.12.3.128 could be a host or a subnet paired with 121.12.0.0 leads rip to think what?
  - convergence is slower (minutes sometimes) AND
  - not as scalable as OSPF - can't aggregate as well
    - hop count max is small (not really important)

not quite concluded

- cons, cont.
  - metric notion overall not flexible
    - **cannot deal with different link types**
  - not so hot with complex topologies; e.g, smart setup of multi-homed (not transit) A.S.

almost the end, really

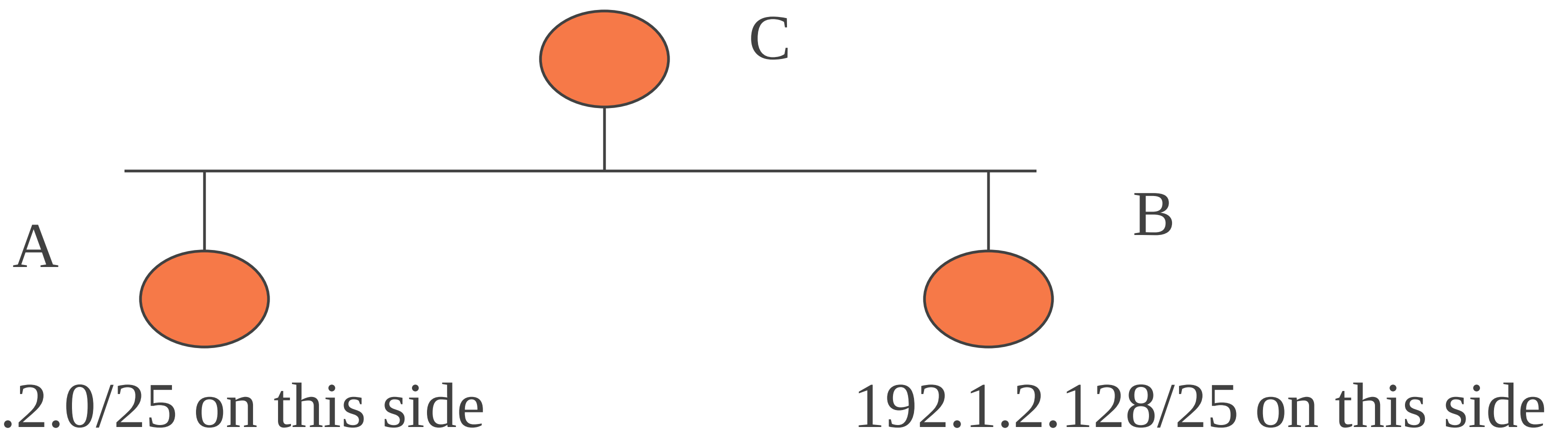
- Cisco has considered DV not a bad technology
- IGRP has composite metric, but still classful
  - RIP++
- EIGRP a “D/V” protocol with == complexity and features to OSPF
  - classless too

almost ... !

- really important cons
  - RIP v1, classful (OK so use V2)
  - hop-count metric brain-damaged
    - » heterogeneous links REALLY likely
    - » 10BASE, 100/1000 Ethernet
- may be OK for feeding info to hosts
- routers SHOULD definitely ignore host RIP suggestions
  - remember “routed -g” ...

1 study question - what to do with RIP and this network?

we own routers A, B, our ISP has C
we have IP address space 192.1.2.0/24, and A uses 192.1.2.0/25,
B uses 192.1.2.128/25, we use RIP to talk to C, what do we say?



1. what happens if we use RIPv1?
2. RIPv2?